FINAL REPORT
under
AASERT N00014-93-1-1216

Training in Research and Construction of Secure Distributed Real-Time Systems

June 25, 1993 — October 31, 1996

Principal Investigator: David C. Luckham

1 Abstract

We had proposed to augment current ARPA and AFOSR funded research and technology projects by supporting additional graduate students, one from ARPA and one from AFOSR and their computing equipment. These students were trained on these DoD research projects in the areas of design and implementation of specification and prototyping languages for system architecture. The students were trained in (1) specification and prototyping of architectures for avionics systems, simulation systems, and other time-critical systems, (2) methods of testing actual products for conformance to architectural standards, and, (3) design and implementation of support tools for simulation and verification of such systems.

The proposal was based on our ongoing research activities on event-based architecture definition languages and support environments for system design, specification, and prototyping. Our ARPA sponsored research involved efforts on the Prototech and DSSA programs (ARPA N00014-92-J-1928), and an effort under the Program Composition program (ARPA N00014-93-1-1335). Our AFOSR efforts in avionics systems took place under AFOSR grant AFOSR-91-0354A.

Students interacted with industrial partners like TRW and Sun, learned first-hand about specific systems projects currently on-going in industry, and worked with industrial scientists and engineers to apply our languages and tools to various projects.

2 Background on Research Projects

2.1 The Anna Specification Language

This project was funded by DARPA through contract number N-00039-91-C-0162.

Anna [?] (**ANN**otated **A**da) is a language extension of Ada [?] to include facilities for formally specifying the intended behavior of Ada programs. Anna was designed to meet a perceived need to augment Ada with precise machine-processable annotations so that well established formal methods of specification and documentation can be applied to Ada programs. Anna is based on first-order logic and its syntax is a straightforward extension of the Ada syntax.

Tools for the Anna specification language [?] have been under development at Stanford University since 1982. These tools are written completely in Ada and include a parser, a semantic checker, a specification analysis tool [?], a transformation tool that converts Anna specifications to Ada checking code for run-time testing [?], and a run-time debugger that can be loaded with transformed Anna programs. Other by-product tools of the Anna project include a parser generator, a Prolog interpreter, and an interactive Anna tutorial. All these tools can be considered advanced prototypes and have been distributed to over 100 sites worldwide.

We have developed schemes to perform the Anna checks concurrently on separate processors [?]. This results in a *permanent self-checking* program, which will perform run-time consistency checks every time it is used. To reduce the overhead of run-time consistency checking, the Anna specification transformer generates checking code that runs concurrently with the underlying Ada program.

Another accomplishment has been the generation of run-time checking of algebraic specifications [?]. This involves monitoring the execution of the underlying program using a theorem prover. The theorem prover determines when checks need to be performed on the underlying program.

2.2 RAPIDE Architecture Definition Language/System

This project was funded by DARPA through contract number N-00014-92-J-1928, and by AFOSR through contract number AFOSR-91-0354A.

The Rapide Language/System project [?] was a joint undertaking with TRW as our industrial partner. Our group was part of a 8 team DARPA Prototech community involved in the process of designing a language for prototyping highly concurrent and time-critical systems. The Rapide language and toolset is implemented and tested on industry problems such as

1. SUN Sparc Version 9 64 bit instruction set architecture (in collaboration with SUN MicroSystems and Sparc International),
2. X/Open Distributed Transaction Processing Model,
3. IBM F16 Flight Simulator, in collaboration with the IBM Stars effort,
4. IBM/Loral ADAGE Avionics systems, in collaboration with TRW.

Rapide is a modular language with sophisticated specification facilities drawing from our experiences with languages such as Anna, TSL, and VAL/VHDL. The *Rapide* execution model is based on pattern invoked processes. We developed tools such as type checkers and graphical editors/animators for Rapide–1.0. These tools are being applied to establish scalability to industry scale problems/systems.

The language design effort defined a framework within which the following four language components interact: (1) the type system; (2) the executable system; (3) the architecture definition language; and (4) the formal constraint specification language.

The type system is based on ML. Extensions include a module declaration capability, and subtyping rules [?]. The executable system is based on pattern invoked processes and defines a notion of distributed time. The specification language is based on our earlier specification languages such as Anna, TSL, and VAL. The specification language also includes pattern based constraints. The architecture definition language incorporates new features developed under our ARPA and AFOSR efforts for defining large scale system architectures.

3 Work Done and Accomplishments

We had proposed to augment current DARPA and AFOSR funded research and technology training projects by supporting graduate students and their computing equipment. Three graduate students — John Kenney, James Vera and Alvin Cham were supported as part of this project.

These students were trained on research projects in the areas of design and implementation of specification and prototyping languages, and their immediate application to (1) construction, simulation and analysis of architectures for avionics and time critical systems, and (2) prototyping of real-time distributed industrial systems.

Given the nature of the projects we undertook, and that we have extensive collaboration with both industry and university group in DoD related projects, these students had the opportunity to interact extensively with a diverse group of DoD sponsored scientists and engineers, and learned first-hand about specific systems projects currently on-going in industry.

Students were supervised in their activities by senior members of the Program Analysis and Verification Group. Students were required to attend our weekly group meetings during which technical discussion of a general nature took place. They also met with their supervisors for detailed discussions at least once a week.

The students were initiated to our research activities with small projects that involve gaining familiarity with the tools already developed in the group :

- Developing syntactic and semantic descriptions of Rapide
- Some basic overload resolution and type checking algorithms
- Prototyping real-life systems in Rapide in collaboration with TRW and other industrial companies. These systems include a Sun SPARC V9 chip.
- User-interfaces for building prototypes graphically using Rapide as the target language
- Implementation of partial order determination algorithms [?] for Rapide

- In-depth modelling of industrial simulation and avionics systems.
- Development of tools for animation of Rapide simulator results.
- Prototyping of transactions processing systems.

Students were asked to participate in one or more of the various subgroup activities. Some of these subgroups activities were:

- Executable language design for Rapide
- Specification language design for Rapide
- Flexible semantics checking tool development
- Branding technology for testing conformance of actual industry products to architectural standards; this will be based upon advanced Rapide constructs for event pattern mappings between systems in various languages.